بايثون للمبتدئين: من الصفر إلى أول مشروع

شرح عربي مبسط + كود إنجليزي واضح + تطبيق عملي



المشروع النهائي: مدير المهام اليومية (CLI)

تأليف: ياسين

KSIEB LIPP

الفهرس

الفصل الأول : كسر الرهبة

ماذا يعني هذا ، يا أخي الكود عندما نسمع عنه يتبادر الى ذهننا شيئ سحري من وحي الخيال ، لكن إنما هو مجرد منطق تضيف له انت افكار ليكبر ،هذا هو
على كمبيوتر حمل vs code وايضا حمل بايثون
لديك لينيكس؟ جيد على الطرفية (Terminal) حمل nano أو vim مع بايثون
ماعندك بيسي ، مش نهاية العالم ، عندك تلفون ضعيف حمل trmux وعليه حمل nano حتى لو رام 1 راح يشتغل
شفت لايوجد سحر مافي كمبيوتر 1000$ ، الان لايوجد أعذار الأن لا اريدك ان تفهم شيئ اريدك ان تفتح محرر اكواد [هناك آلاف الفيديوهات لكيفية تحميل وتثبيت كل ماذكرته على اليوتيوب مجانا!] ، وأن تكتب التالي فقط لا شيئ اخر وتشغله لاتحاول الفهم هذا الفصل كي تشغل كود لأول مرة وترى انه امر بسيط!!.

```
print ("I am learning python !!!")
```

هذا وبس رح يكتب على الطرفية I am learning python!!!
ويمكنك كتابة ماتريد شرط ان يكون بين قوسين وعلامتي التنصيص(" ......")

الفصل الثاني : أنواع المتغيرات Data type

في لغة بايثون انواع كثيرة ولكن مايميز بايثون يا عزيزي القارئ هو انها تعتني بك حرفيا ، لست محتاجا لذكر اسم المتغير لتخزينه يكفي ان تكتب امر ببساطة وبايثون تحدد النوع عنك

النوع الاول int
هو نوع مختص بتخزين الاعداد الصحية والسالبة ونكتب
x=1
x=-1
هذا فقط
النوع الثاني float
هو نوع مختص في تخزين الاعداد العشرية او كسور كانت سالبة او موجبة ونكتب
X= 1.1
X=-1.1
النوع الثالث str
وهو نوع مختص في تخزين النصوص داخل متغيرات ونكتب
x= "hello world"

توقف هنا للحظة ⛔⚠️

هنا تقع بعض اخطاء المبتدئين
لنناقش واحدة هنا
علامات التنصيص او كما تعرف بعلامات الاقتباس مهمة وتغير المعنى
بدون علامة تنصيص مع نص او جملة او حرف الي ناوي تخزنه يعني error ❌
الصح ✅ x= "hello"
الغلط تحط علامة تنصيص مع int او float يعني لو سويت "5"+"5" = "55" لانه يعتبرها نص و نصوص تجمع بالحاق اخر كلمة من اولى باول كلمة من الثانية ❌
الصح تكتب بدون علامة تنصيص 5+5=10 ✅
يمكنك تسمية المتغير كما تريد وليس حكرا على x
x=1
y=10
myname="yacine"
الشروط للتسمية
عدم البداية برقم ❌
يعني ممنوع 1x=1❌
الصحيح x1=1✅
بايثون حساسة للأحرف⚠️
لاتسميه x ثم تستدعي X والعكس صحيح ❌ ولا تضع مسافة
سميته x صغيرة استدعه بها ولو سميته X كبيرة استدعه بها
الحالة تفرق ✅ وايضا بدل المسافة حط _ مثل my_name
النوع الرابع bool

وهو متخصص في تخزين قيمتين منطقيتين فقط

x= True

z= False

لا تكتب true أو false ❌

الصحيح True أو False ✅وسنتعرف استخداماتهم في حلقات تكرارية

النوع الخامس list

هو متغير لتخزين عدة قيم في قائمة ونكتب

x=["yacine","mohammed","10"]

تنادى ب index وهو من الصفر يبدا اي عند منادات القيمة صفر يخرج اسم yacine والقيمة واحد Mohammed وهكذا

النوع السادس dictionary

القاموس وهو طريقة افضل لتخزين المعلومات ونوع متقدم من قوائم حيث تنادى بالاسم بدل index

```
x={
"Name" : "yacine",
"age":18
.....
}
```

وهكذا

من الاخطاء الشائعة

منادات القاموس بالindex مثل القائمة والعكس صحيح❌

لازم علامات تنصيص للنداء مثل name ❌"name"✅

الفصل الثالث: استخدام تطبيقي للمتغيرات وتعلم امري الطباعة والاستقبال

لعل المتغيرات اهم شيى، لانها تخزن الكلمات والحروف وارقام
مثال عملي بسيط كلمة المرور تستخدم متغيرات لتخزين الاسم وكلمة المرور
امر الطباعة هو نفسه الامر الذي استعملناه في البداية
وتترجم من الانجليزية الى طباعة print
وهو امر المستخدم لجعل البرنامج يرجع او بمعنى اصح يطبع قيمة
مثل

```
print("I am learning python")
```

تطبع على الشاشة

```
I am learning python
```

أمر الاستقبال هو امر يسمح لنا بادخال معطيات من المستخدم
input
ويستعمل هكذا

```
x=input("enter your age\n")
```

يطبع

```
Enter your age
```

يستقبل من مستخدم عمر الذي يدخله وعندما يضغط المستخدم enter يخزن عمره في متغير x

ملاحظات

\n

تستخدم للنزول سطرا ويحبذ استخدامها دوما عند نهاية اي امر للطباعة او استقبال

مثال

```
print("my name is yacine \n I am 18 years old \n")
```

يطبع

```
my name is yacine
I am 18 years olde
```

وينزل سطر اخر

وهذا مايسمى بالاستعمال الجيد

ملاحظة اخرى يمكنك كتابت بين القوسين اي رسالة تظهر للمستخدم ما يجب ادخاله

يجب تضمين الاستقبال داخل متغير لحفظه

التطبيق العملي للكود

كود كتابة مستقبل اسم مستخدم وكلمة المرور وطباعتها للمستخدم

انسخ وجرب

```
print("|welcome| \n")
user_name=input("enter your user name \n ")
password=input("enter your password now \n")
print("your name is "+user_name+" and your password is "+password)
```

اولا يطبع
|welcome|
ثم ينزل سطر ويطبع
enter your user name
فيستقبله وبمجرد ضغط المستخدم enter يخزن في المتغير المخصص له و في هذه الحالة هو user_name
فينزل سطرا اخر ويطبع
enter your password now
ثم ينزل سطرا اخر ويطبع
your name is يكتب اسم هنا and your password is
ثم يكتب الباسورد بنفس السطر
تنبيه في الباسورد الحقيقي لا نستخدم امر طباعة الاخير هو حاليا لتعلم فقط ورؤية كيف يخزن الكود القيم مؤقتا
ومن الاخطاء الشائعة عدم وضع \n بالنهاية مما يعل كل شيئ في سطر واحد
هنا التخزين ليسا نهائيا بل مؤقت بمجرد اعادة تشغيل الكود اختفى
سنتعلم لاحقا التخزين الدائم
وهناك طريقة اسهل للكتابة المتغيرات
نستخدم f فنكتب
print(f"your name is {user_name}")
ونستخدم { نكتب اسم المتغير هنا} وهذا ليعرف البرنامج انه متغير وليس بنص

الفصل الرابع : التعليق #

التعليق مفيد جدا ويسهل تصدير الكود وتعديله مثلا لو كنت تعمل مع مجموعة تحتاج ان تشرح الكود لهم ليكملوا ، ببساطة ضع تعليق ،اذا ماهو التعليق ؟ هو كلام تضعه باي لغة تريد بعد # ولا ينفذ ابدا ولا يقرئه التطبيق اصلا وهو مفيد لتسهيل القرائة وتعديل وتذكر عندما تعود للكود بعد مدة طويلة وتريد التذكر لماذا هذا هنا

```
#that my name
print("yacine")
```

الكود يطبع yacine و هذا كل شيئ
كانه لم يرى التعليق اصلا
الرجاء الانتباه وعدم كتابتها قبل امر يالخطأ
لو أخطأت فلا تتفاجأ بأنه لم يعمل

الفصل الخامس : العمليات في بايثون
الفرع الاول : العمليات الحسابية

العمليات الحسابية في بايثون كالحياة الواقعية ورياضيات تماما

لدينا الجمع
```
print(5+5)
```
يطبع هذا الكود 10
لو اخطأ ووضعت علامات تنصيص فحسب خطأك
```
print("5+5")
```
يرى الكود هذا كنص فيطبع
5+5

ولدينا عملية الطرح -
```
print(1-1)
```
يطبع هذا الكود
0

العملية الثالثة هي الضرب *
وتكتب بالنجمة *
```
print(1*1)
```
وهذا الكود يطبع
1

العملية الرابعة القسمة ونرمز لها ب / وانتبه ولا تضع \ فيظهر خطأ

```
print(1/2)
```

وهذا الكود يطبع

0.5

والأخيرة هي حاصل القسمة

تقوم هذه العملية باظهار باقي القسمة الإقليدية ورمزها %

```
print(5%3)
```

هذا الكود يطبع

2

لأن ناتج القسمة 5على 3 هو 1 والباقي هو 2

الفرع الثاني : العمليات الشرطية

وهي عمليات تستخدم في الشروط وسنعرفها لاحقا

العملية الاولى: أكبر من وتكتب > وتستخدم للتحقق اذا a اكبر من b فنكتب a>b

العملية الثانية :أصغر من وتكتب < وتستخدم للتحقق اذا كان a اصغر من b فنكتب a<b

العملية الثالثة: أكبر من أو تساوي وتكتب >= وتستخدم للتحقق
اذا كان a اكبر من b او يساويه اي اذا كان اكبر او كانا
متساويان فالنتيجة في كلتا الحالتين صحيحة فنكتب a>=b

العملية الرابعة : اصغر من او تساوي وتكتب<= وتستخدم
للتحقق اذا كان aاصغر او يساوي b فنكتب a<=b

العملية الخامسة : المساوات وتكتب == باثنين علامات مساوات
انتبه اذا كتبت a=b فانت تخزن قيمة المتغير aفي المتغير b
الصحيح ان تكتب a==b وبهذا انت تتحقق اذا a يساوي b

العملية السادسة: عدم المساوات وتكتب != وهي تستخدم
لتحقق اذا a لاتساوي b فنكتب a != b

الفرع الثالث : العمليات المنطقية

لدينا ثلاثة
هم عمليات تستخدم مع الشرطية لتحديد بعض الخصائص

العملية الاولى : and وكما هي بالانجليزية تعني [و] او [مع] وهي تستخدم للتحقق اذا ما تحقق شرطان معا ويجب ان يكونا مع او الشرطان غير متحققان لنعتبر c تعني شرط عند قول c1 and c2 يعني اما ان يتحقق الشرطان معا او يعتبر الشرط غير محقق

العملية الثانية : or وكما هي بالانجليزية تعني [أو] وهي تستخدم للتحقق اذا ما تحقق احد الشرطين ويكفي تحقق احد الشرطين فقط لتعتبر صحيحة وعندما نكتب c1 or c2 فهذا يعني اذا تحقق واحد فقط مثلا c1 ولم يتحقق c2 فان or تعتبره صحيحا

العملية الثالثة : not وكما هي بالإنجليزية تعني [لا] وهي تستخدم للتحقق اذا كان الشرط غير محقق ونكتب not c1 اي يجب ان لا يتحقق الشرط c1 لتعتبر not صحيحة

عزيزي القارئ هذا يعتبر مجرد شرح نظري اقرئه وحاول فهمه قدر الامكان لكن من خبرتي الشرح النظري لا يغني شيئا لذى في الفصل التالي ساشرح الشروط وادخل العمليات فيها لقيام بمشروعات هي الالة الحاسبة ،و روبوت تحقق اذا يسمح لك بالدخول ،و عملية تسجيل الدخول في المواقع لفهم هذا تطبيقيا فاستمتع ولا تنسى الكتاب مجاني تماما على صفحتي على الفيسبوك ومسجل باسمي في ارشيف الانترنت

الفصل السادس : الشروط

الشروط هي عملية تحقق تتم عبرها تعليم البرنامج ماذا يفعل ولماذا

الشرط الاول : if هو الشرط الأول والأساسي وبه لن تكون هناك شروط ونستخدم للتحقق اذا الشرط الذي يليه صحيحا

```
# نكتب الشرط وهذا تعليق وكما تعلمنا البرنامج يعتبرها غير
# موجودة ويجب وجود هاشتاق لكل سطر
if 1>0 :
#ننزل سطر ونترك اربعة فراغات دوما وفي محررات تترك
#تلقائيا او بضغط على TAB في لوحة مفاتيح الكمبيوتر
    print("yes")
```

و هذا الكود يطبع

yes

واذا لم يكن محققا مثل

```
if 1>2:
    print("yes")
```

هذا لا يفعل شيئا لان الشرط غير محقق فيتجاوزه

ملاحظة يجب وضع[:]مع نهاية كل شرط وترك اربع مسافات
للكود المرجو تققه والا تنفذ الكود بلا تحقق

: نكتب الشرط if
نترك اربعة مسافاة اونضغط TAB ونكتب الكود المرغوب
في تحققه بتحقق الشرط
تذكر عدم ترك المسافات يحقق الكود بلا تحقق
ولمتابعة كتابة بعد انتهاء من الشروط لا تكتب مسافة
if الشرط:
الكود للشرط
باق الكود

الشرط الثاني :elif هو شرط نريد التحقق منه اذا ما لم يتحقق
الشرط ويمكننا كتابة كما نريد منه اي عدد الذي تريد
ملاحظة اذا تحقق if يلغي التحقق من البقي
يمكننا كتابة كما نريد منها ويتم التحقق منها بالترتيب elif
اذا تحقق elif واحد تلغى البقية

```
if 1<0:
    print("1<0")
elif 1>0:
    print("1>0")
```

هذا الكود يطبع
1>0 ويتخطى التحقق الاول لأنه غير محقق

الشرط الثالث: else وهو يكتب بدون شرط معه فقط :else
وهو يسمى الشرط الافتراضي اي انه يتحقق عندما تفشل كل الشروط

```
if 1==0:
    print("1==0")
elif 1<0:
    print("1<0")
else:
    print("sorry")
```

هذا الكود يتحقق من اذا كان 1 يساوي 0 فيجده خاطئ فيتخطاه ، ثم يتحقق اذا كان 1 اصغر من 0 فيجده خطأ فيتخطاه ، لا يجد شرط اخر فينفذ الشرط الافتراضي ويطبع sorry

ملاحظة لا تضع شرط else الاول او الوسط فهو دوما محقق وتنفيذ يتم بالترتيب

لو تحقق شرط قبل else فان else لاينفذ

دائما تبدأ ب if ويكتب مرة فقط في كل جملة شروط

عند وجود if اخرى فانها تعتبر ان هذه جملة شروط منفصلة تكتب مرة فقط else

مثال تطبيقي
سنحول هذا الى مثال تطبيقي فيه كل مادرسنا لحد الان
الالة الحاسبة

نقوم اولا بطباعة ترحيب للمستخدم

print("welcome to my calculator \n")

لا ننسى نزول سطر كي لا نتسبب في تداخل الكلام

first_num= float(input("what's your first number \n"))

#هذا يستقبل اول رقم ويخزنه في متغير
كتابة float قبل ادخال ووضع ادخال بين قوسين تجبره
#على ادخال بقيمة عشرية وليس نص
operation= input("which operation your want \n")

هذا يستقبل العملية من من المستخدم ويخزنها في متغير

second_num=float (input("what's your second number \n"))

#هذا يستقبل الرقم الثاني ويخزنه في المتغير

الان يا عزيزي القارئ وصلنا الى نهاية الجزئ الاول
هكذا استقبلنا كل مانحتاج من مستخدم
الان يجب ان نطبع النتيجة ولكن اولا نتحقق من عملية
كن فخورا ستعلم الة كيف تحسب
تنويه العملية تخزن كنص وليس رقم لذى وجب وضع "."

```
if operation == "+":
print(f"result is {first_num +  second_num}")
```

بسبب كون ادخال المستخدم للعملية كنص نظطر لذلك

#نتحقق يدويا من العملية المدخلة ثم باستخدام f

التي تعلمناها نضع الرقمين عبر متغيراتهما ونضع العملية

التي تحققنا منها داخل {..}

```
elif operation == "-":
print(f"result is {first_num -  second_num}")
```

اذا لم يكن ادخل جمع سيتحقق اذا ادخل طرح

```
elif operation == "*":
print(f"result is {first_num * second_num}")
```

اذا لم يكن ادخل لا جمع ولا طرح يتحقق اذا كان ضرب

elif operation == "/" and second_num != 0:
print(f"result is {first_num / second_num}")

في حال لم تكن ايا من سبق يصل الى هنا

وهذا تطبيق ممتاز للعمليات

يتحقق اذا كانت الغملية هي القسمة / ثم تتدخل and
يتحقق اذا كان لا يساوي الصفر لان القسمة على صفر خطأ
الand تقوم بتحقق من شرطين ان لم يتحقق احدهما
تلغي كل شيئ

elif operation == "%":
print(f"result is {first_num % second_num}")

واذا لم تكن اي من السوابق ياتي الى هنا لينفذ عملية باقي
القسمة

else:
print("sorry check, what you write ")
واذا لم تكن اي عملية يعني خطأ من المستخدم فيطلب منه
#التحقق
مثال تطبيقي أخر

هو روبوت التحقق اذا يسمح لك بالدخول

```
# نطبع ترحيب
print("|welcome|\n")

# لنعتبر ان هذه حفلة ولا يحق لك الدخول الا بدعوة وسن 18

age=int(input(" what's your age\n "))
# هذا يستقبل العمر

invite=input("did you have invite yes or no\n")
# هذا يستقبل تحقق اذا لديه دعوة
# نبدأ التحقق

if age >= 18 and invite == 'yes' :
    print("welcome sir")
# اذا توفرت لديه دعوة وهو اكبر من 18 او عمره 18 يرحب به

elif age < 18 or invite != 'yes' :
    print("sorry sir")
# اذا كان عمره اقل او لايمتلك دعوة او اثنان معا يرفضه
```

مثال اخر هو تاكد من كلمة المرور

نطبع ترحيبا بالمستخدم

print("|welcome |")

نخزن كلمة المرور والمستخدم مسبقا
password ="admin123"

باسوورد ضعيف لاتستخدموه للشبكة

user_name ="admin"

نستقبل المدخلات
user= input(" enter user name ")
P
Pass = input(" what's your password ")

نتحقق منها
if password == Pass and user_name ==user :
print("login successfully")
اذا تحققت كل الشروط اقبله

```
else:
print(" user name or password incorrect ")
# والا ارفضه
```

ملاحظة لايمكنكم استخدام pass كحرف صغير اول لانها مجوزة في بايثون لذا استخدمت Pass بحرف كبير بايثون حساسة لحالة احرف

الفصل السابع : التخزين الدائم في الكود عبر ملف خارجي

كما رأينا سابقا في كسر الرهبة من طباعة والاستقبال كيف اخذنا كلمة المرور وطبعناها ، وهذا خطأ ولكن كان مثال شرح ، وتتذكرون عند تحقق بالشرط من كلمة مرور في متغير داخل الكود ، هذ الطريقة لم تعد مستعملة الان التخزين يتم عبر ملف خارجي لكلمة المرور و يتم التحكم فيه عبر الكود، اين يكون هذا الملف داخل التطبيق الذي تم تجميعه، لا اريدك يا أخي ان تتعب نفسك الأن بهذا ، كل ما اريدك ان ان تفعله ان تطبق مع الكود فتجد انك انشأت ملف خارجي تخزن فيه كلمة المرور وتتحقق بها من الكود ، ولن تعرف فرحة بدايتك للاحتراف، اجل فهذه هي الطريقة الصحيحة المتبعة الفرق انهم يجمعونه وانت تختبره وجمع يا أخي سيكون نفس الكود ولكن فقط تستخدم مجمع وياخذ ثوان ،ثوان بينك وبين الاحتراف هذا فقط ، لذا كن فخورا

ملاحظة في طرق الاحترافية يتم تشفير الملف قبل تجميع الكود

نقل الملف من المجلد الذي فيه الكود يجعل الكود لا يتعرف عليه

لنبدأ اولا انشاء الملف والكتابة فيه

لانشاء الملف نكتب

```
open("password.txt", "w")
```

لنشرح الكود

open

هو امر انشاء الملف نستخدمه مثل امر الطباعة او الاستقبال

password.txt

الكلمة الاولى password هو اسم الملف المرغوب بإنشائه وهو اختياري حسب رغبتك

الثانية .txt هو امتداد الملف وهو اختياري حسب رغبتك ولكن لتخزين الكلام نستعمل .txt

بعد الفاصلة وعلامتي تنصيص هذا هو المهم

w

تعني write اي كتابة وهي لها عملان اما تنشئ ملف وتكتب به أو اذا كان الملف موجودا تحذف القديم وتكتب الجديد

تسمى w بالوضع وهناك اربع

w

شرحناها

a

وهي التي انصح بها وتعني append اي اضافة ولها دورين اما تنشئ ملف ونكتب فيه ، واذا كان الملف موجودا مسبقا تضيف له بدون ان تحذف المحتوى القديم

r

وهو يعني read اي قراءة وله دور اي قرائة ملف باسم الذي تكتبه ، اذا لم يكن موجودا تظهر خطأ

x

وتعني create ولكنهم وضعوا رمز x وهو لانشاء ملف فقط ، لو الملف موجود سيظهر خطأ

تذكر بايثون حساسة للاحرف اي اذا سميت ملف ب Data.txt لا تأتي وتحاول قرائته بأمر

open("data.txt","r")

سيظهر خطأ أن الملف غير موجود وهذا لان الحرف في Data.txt الأول كبير وليس صغير

اذا لنقم بتطبيق عملي سريع لكتابة كلمة مرور والتحقق منها وتعلم تطبيقي هو أفضل

```
# نستقبل من المستخدم كلمة المرور
password=input("enter your password ")
#ننشئ ملف
file =open("saved_password.txt","w")
# استخدمنا w لانه ملف باسوورد وسيتم استبداله كل مرة
# الكلمة في البداية هي تسمية للكود
# بدل كتابة كود كل مرة لتحديد الملف الذي سنكتب فيه
# نسمي الكود file باستخدام =
```

هذا يعني انه كلما كتبت f انت تستدعي الكود
ويمكنك التسمية كما تشاء
file.write(f"{password} \n")

رايت لولا تسميته file لاعدنا كتابة الكود هنا
بسبب تسمية file كل ماكتبناه هو file.write
وذلك هو امر الطباعة داخل الملف ويكتب write
هو ليس نفسه w فذلك وضع و الكلمة الكاملة بعد نقطة امر
يستخدم الامر مع وضعين wو a عادي

يجب الان اغلاقه ووضعه في وضع القراءة للتحقق
file.close()
الان فتح وتسمية بنفس اسم او يمكنكم التغيير
file =open("saved_password.txt","r")
وضع القراءة هو r كما تعلمنا سابقا

الان التحقق
data= file.read()
نستخدم .read وهو امر القراءة لتخزين في متغير
سميت المتغير data ، نحتاجه لان بايثون لاتقارن من ملف

الان نطلب من المستخدم ادخال كلمة المرور

```
check=input("enter the password \n")
```

نستقبل ادخال المستخدم ونخزنه في متغير

```
if data.strip() == check:
    print(" welcome sir ")
else:
    print("no sir that isn't the password ")
```

احنا ايش زدنا ، ملاحظ صح .strip() ، هذي لان التخزين كلمات المرور احيانا يضيف فراغات بالغلط وبايثون حساسة لازم حرف حرف ، ف .strip() تلغي اي فراغات انضافت بالغلط

لاحظنا شيئ، كل مرة لازم close() ، لو تم نسيان اقفال الملف لن يتم حفظ وتخزين المعلومات
بايثون حطت حل هو with open
يعني تكتب

```
with open("saved_password.txt","w") as file:
```

و الفرق هو اضافة with في البداية
وبدل كتابة =file نكتب as file مع الاخير
وهكذا تقوم بعمل نفس الشيئ بدون ان تضطر للاغلاق كل مرة

الكود المحدث

```
# نستقبل من المستخدم كلمة المرور
password=input("enter your password ")
#ننشئ ملف
with open("saved_password.txt","w") as file :
# استخدمنا w لانه ملف باسوورد وسيتم استبداله كل مرة
# الكلمة في البداية هي تسمية للكود
# بدل كتابة كود كل مرة لتحديد الملف الذي سنكتب فيه
# نسمي الكود file باستخدام =

# هذا يعني انه كلما كتبت f انت تستدعي الكود
# ويمكنك التسمية كما تشاء
file.write(f"{password} \n")

# رايت لولا تسميته file لاعدنا كتابة الكود هنا
# بسبب تسمية file  كل ماكتبناه هو file.write
# وذلك هو امر الطباعة داخل الملف ويكتب write
# هو ليس نفسه w فذلك وضع و الكلمة الكاملة بعد نقطة امر
# يستخدم الامر مع وضعين wو a عادي

# لا  يجب الان اغلاق
```

الان فتح وتسمية بنفس اسم او يمكنكم التغيير
with open("saved_password.txt","r") as file:
وضع القراءة هو r كما تعلمنا سابقا

الان التحقق
data= file.read()
نستخدم .read وهو امر القراءة لتخزين في متغير
سميت المتغير data ، نحتاجه لان بايثون لاتقارن من ملف

الان نطلب من المستخدم ادخال كلمة المرور

check=input("enter the password \n")

نستقبل ادخال المستخدم ونخزنه في متغير

if data.strip() == check:
print(" welcome sir ")
else:
print("no sir that isn't the password ")

الكود المحدث احسن اكثر احترافية واكثر حماية من الاخطاء

الفصل الثامن : التكرار

هناك بعض الامور تحتاج تتكرر باستمرار ، لنقل نريد انك تريد طباعة اسمك خمس مرات
لن تكتب

```
print("yacine")
print("yacine")
print("yacine")
print("yacine")
print("yacine")
```

هذا طويل ، لو قلنا 1000 هذا كثير ، ولكن بايثون تعتني بك فقد وفرت لك حلقات تكرارية

الفرع الأول : التكرار while

هذا هو اشهر التكرارات في بايثون أو في أي لغة ، يسخدم لجعل التطبيقات تعمل أو لتكرار شيئ لأجل
مثال تكرار اسم

```
# اولا نضع متغير قيمته 0
i = 0
while i <5:
    print("yacine")
    i=i+1
```

شرح التكرار لأجل
نضع متغير قيمته صفر لأن بايثون تبدأ من الصفر
نضع شرط اصغر من خمسة لانه يبدأ من الصفر
فلطباعة خمس مرات نبدأ من صفر الى اربعة وهكذا نطبع خمسة
ولو اردنا وضع اصغر او تساوي خمسة نضع المتغير بقيمة واحد

ننزل ونترك اربعة فراغات مثل شرط تماما
نكتب امر طباعة
ننزل ونترك اربعة فراغات اخرى ونكتب اسم المتغير يساوي المتغير نفسه ونضيف له واحد لكي تزيد قيمة المتغير دوما حتى تصل الى قيمة الشرط ويتوقف
بلا i=i+1 سيدخل في حلقة غير متناهية الى الابد
او الى ان يتوقف جهازك بسبب حرارة المعالج من تكرارات الامتناهية
وهناك صيغة مختصرة هي
i +=1
هذا هو الاختصار
لا تنسى نقطتين زي شرط
يعني نقطتين و تنزل اترك اربع مسافات زي الشروط if

شرح التكرار وفق شروط

تتذكرون نوع المتغيرات bool ، المتغيرات المنطقية ،True False هنا نستخدمها

لتشغيل التكرار الابدي من اجل ايقافه وفق شروط
لانريد تكرار ابدي لذا هناك شروط توقفه
امر الاستقبال و امر الطباعة يوقفانه مؤقتا
امر break ايقاف فوري

تطبيق على كلمة المرور

من الفصل السابق

```
# نستقبل من المستخدم كلمة المرور
password=input("enter your password ")
#ننشئ ملف
with open("saved_password.txt","w") as file
file.write(f"{password}\n")

# الان فتح وقراءة
with open("saved_password.txt","r") as file
# وضع القرائة هو r كما تعلمنا سابقا
```

الان التحقق
data= file.read()
نضع حلقة غير متناهية
while True :
الان نطلب من المستخدم ادخال كلمة المرور
نضعه هنا لاننا نريده اذا اخطأ يعود ليجرب مجددا
check=input("enter the password \n")

نستقبل ادخال المستخدم ونخزنه في متغير
نترك اربعة فراغات لتضمين الشرط داخل الحلقة
if data.strip() == check:
نترك 8 فراغات لتضمين الكود داخل الشرط وداخل التكرار
print(" welcome sir ")
break
لو كان صح نقوم ب كسر عبر break لايقاف الحلقة
else:
print("no sir that isn't the password ")
هنا يعيده لكي يدخل كلمة المرور من جديد في حلقة

بدون ال break تحت الاجابة الصح رح يضل يسأل للابد

نصيحة جرب الاكواد لتتعرف على ما يحصل وتكتسب الخبرات

مقولة شهيرة : " القرارات الصائبة تأتي من الخبرة ، والخبرة تأتي من قرارات خاطئة "

الفرع الثاني : التكرار for

ال for تختلف عن while، فبينما while تستخدم للتكرار لاجل او وفق شروط ، تستخدم for مع قوائم والملفات وفق شروطها الخاصة
مثال تطبيقي

```
# ننشئ قائمة
x=["yacine","Hossam", "aly","abd almoaise"]
# نبدأ التكرار
for i in x:
print(f"{i} \n")
# التكرار for
# المتغير i هو متغير يستخدم مرة واحدة وكل مرة يعوض
# بعنصر من القائمة
# ال in هو تحديد ، وهو يسحب من القائمة و يضع في I
# القائمة هي x
# نضع نقطتين ونترك اربع مسافات
```

الكود سيطبع
yacine
Hossam
aly
abd almoaise
لولا n\ لكتب
yacineHossamalyabdalmoaise
لهذا نستخدم n\
وايضا for يتوقف تلقائيا ليس ك while يجب ايقافه
ولكن يستخدم مع قوائم وملفات فقط
ولكن هناك امر
i
هو متغير مؤقت ، نستخدمه للطباعة فقط ، لذا يفضل استخدام متغير باسم _ وهو متغير لايحفظ ويستخدم مع for فقط ،لا تستخدمه كمتغير عادي اطلاقا

نكتب

```
x=["yacine"," Hossam", "aly","abd almoaise"]
for _ in x:
print(f"{_} \n")
```

وهناك استعمال اخر مشترك مع while، وهو rang اي الطباعة المتعددة

مثال

```
for _ in range(5):
print("yacine")
```

الفصل التاسع: الدوال def
الان هذا هو فصل الختام ورحلة الاحتراف للاساسيات التي يبنا عليها اي كود في بايثون ، والفخر هو شعارنا كالعادة
رأينا من قبل as وقلنا انها تستخدم لاعطاء سطر كود اسم نستدعيه بيه كل مرة
الدالة def هي ان تسمي عدة اسطر باسم تستدعيه به كل مرة وتساعد في جعل الكود منظم
المحترفون يستخدمون الدوال لتنظيم الكود واذا ظهر خطأ يأتي تحسينه بشكل اسهل كونوالكود مقسم
مثلا اله حاسبة للجمع
نفصل الاستقبال والجمع عن بعض بدوال
تكتب def وتسمي الاسم باي اسم وفق نفس قواعد تسمية المتغيرات حيث بدل الفراغ نستخدم _ ولا نبدأ برقم وحالة الحرف تفرق ثم نضيف () ثم : مثل الشروط والتكرار ونترك اربع فراغات
والاستدعاء يكون اسم الدالة متبوعة ب قوسين فقط وحسب ترتيب العمل

مثال تطبيقي

```
# نعرف الدوال اولا ب def
# وننهيها بقوسين ونقطتين وننزل ونترك اربع فراغات
def get_nember():
    global x,y
# نضيف global ونضع المتغيرات وبينعا فواصل
    x=input("first number \n")
    y=input("second number \n")
# هذا يستقبل من المستخدم الاعداد
# بدونها لن تعمل المتغيرات خارجة الدالة
def add():
    print(f"result is {x+y}")
# هذه الدالة للجمع
# نستدعي الدوال لتفعيلها
get_nember()
add()
```

وطريقة الاكثر احترافية للاستدعاء نكتبهم بالاخير تحت main

```
# نعرف الدوال اولا ب def
# وننهيها بقوسين ونقطتين وننزل ونترك اربع فراغات
```

```
def get_nember():
    global x,y
    x=input("first number \n")
    y=input("second number \n")
# هذا يستقبل من المستخدم الاعداد
def add():
    print(f"result is {x+y}")
# هذه الدالة للجمع


if __name__ == "__main__":
# هذا نموذج محجوز ينسخ كما هو
# نستدعي الدوال لتفعيلها
    get_nember()
    add()
# نترك اربع فراغات
```

والان لنتعلم احد اهم المفاهيم

انواع الاخطاء بابايثون

SyntaxError

ZeroDivisionError

NameError

TypeError
ValueError
IndexError
KeyError
AttributeError
IndentationError
ImportError
FileNotFoundError
Logical Error

ياخي لا تعب نفسك وفكك من كل ذا
في طريقة من بايثون الي بتعتني بنا لنقوم بجعل الخطأ لايظهر
وهي try except ورح اشرحها في المشروع

[مشروع مدير مهام بسيط]

```
def show_tasks():
# دالة عرض المهام
    try:
```

الي اتعلمناها قبل شوي تعني جرب

```
with open("tasks.txt", "r") as file:
print("\n--- قائمة مهامك الحالية ---")
print(file.read())
# يطبع مباشرة قائمة المهام
except FileNotFoundError:
# هذا الاستثناء ولو تلاحظون وضعت اسم الخطأ الي يظهر
# هذا ليس عشوائي يجب ان تكون ملمم بكل خطأ ولماذا يظهر
# هنا احتمال ان لا يكون هناك مهام مسجلة فيضهر هذا الخطأ
print("\nلا توجد مهام محفوظة بعد!")
# بدل عن توقف البرنامج يطبع لك انه لا توجد مهام
while True:
print("=== تطبيق المهام اليومية ===")
print("1. إضافة قائمة مهام جديدة")
print("2. عرض المهام المحفوظة")
print("3. خروج")

choice = input("اختر رقمًا: ")

if choice == "1":
```

```
task = input("اكتب المهام التي تريد إضافتها: ")
with open("tasks.txt", "a") as file:
    file.write(f"- {task}\n")
print("تم حفظ المهام بنجاح!\n")
elif choice == "2":
    show_tasks()
elif choice == "3":
    print("بالتوفيق في يومك!")
    break
else:
    print("خيار غير صحيح، أعد المحاولة.\n")
```

البقية نعرفهم، نفعل وضع التكرار لكي يبقى داخل التطبيق و ايضا نضع خيارات نطبعها ويصبح لدينا واجهة بدون رسومات وهكذا يصتطبيقا حقيقيا

ونضع الشروط وفق ماتعلمنا ونترك اربع مسافات للتكرار واربع للشروط وبالتالي 8 لاكود بعد الشروط
ونستخدم الطباعة والاستقبال والمتغيرات والعمليات الشرطية والدوال والتكرار وكل ماتعلمناه

كانت الكتابة شرفا لي لذا قررت ترك الامور المملة للاخر لكي تفهم بعد التجريب ماذا كنت تفعل يا أخي

الفصل العاشر : ماهي البرمجة وكيف تعمل

البرمجة يأخي هي أن تعطي أوامر مخصصة للكمبيوتر لكي يقوم لك بعملل محدد

أليس الكمبيوتر يتكلم ال1 و 0 نعم يأخي تلك لغته ، لغة البرمجة هي لغة وسيطة قريبة للغته ونفهمها نحن كبشر يفهمها الكمبيوتر بعد التجميع او التشغيل المبلشر بتدخل مترجمات تحولها ل 0 و 1 وكلنا كبرت الترجمات قلت السرعة ولكن بايثون رغم كونها لغة مفسرة بمترجم كبير الا انها من اسرع اللغات و ملتهمت المجالات التي تحتاج صيانة دورية كالذكاء الاصطناعي وغيرها لان العالم يتجه لكود سهل التعديل بسبب توفر رقاقات المصغرة التي تعمل بالكفاءة القصوى

عن الكاتب:

قديري ياسين (Guediri Yacine)، طالب بكلية علوم المادة بجامعة معسكر، وشغوف بالبرمجة والتعلم الذاتي في المجال التقني.

يعمل على تطوير مهاراته بشكل مستمر عبر منصة كورسيرا؛ حيث درس لغة C من خلال دورات البروفيسور الراحل Ira Pohl (جامعة كاليفورنيا - سانتا كروز)، إلى جانب أساسيات الاختراق الأخلاقي (EHE) المعتمدة من منظمة EC-Council.

أُلّف هذا الكتاب في لغة بايثون بهدف نشر العلم النافع وابتغاء حسناتٍ جارية ُتفيد كل طالب علم.

حقوق النشر والتوزيع: